

# Optimization of a multi-stage ATR system for small target identification

Tsung Han (Hank) Lin, Thomas Lu<sup>1b</sup>, Henry Braun<sup>c</sup>, Western Edens<sup>d</sup>, Yuhan Zhang<sup>e</sup>, Tien-Hsin Chao<sup>b</sup>, Christopher Assad<sup>b</sup>, and Terrance Huntsberger<sup>b</sup>

University of California, San Diego<sup>a</sup>, San Diego, CA, Jet Propulsion Lab/California Institute of Technology<sup>b\*</sup>, Pasadena, CA, USA, Arizona State Univ.<sup>c</sup>, Butler Univ./Purdue Univ.<sup>d</sup>, California Polytechnic Univ.<sup>e</sup>

## ABSTRACT

An Automated Target Recognition system (ATR) was developed to locate and target small object in images and videos. The data is preprocessed and sent to a grayscale optical correlator (GOC) filter to identify possible regions-of-interest (ROIs). Next, features are extracted from ROIs based on Principal Component Analysis (PCA) and sent to neural network (NN) to be classified. The features are analyzed by the NN classifier indicating if each ROI contains the desired target or not. The ATR system was found useful in identifying small boats in open sea. However, due to “noisy background,” such as weather conditions, background buildings, or water wakes, some false targets are mis-classified. Feedforward backpropagation and Radial Basis neural networks are optimized for generalization of representative features to reduce false-alarm rate. The neural networks are compared for their performance in classification accuracy, classifying time, and training time.

**Keywords:** automatic target recognition, correlation, neural network, small target, false alarm rate.

## 1. INTRODUCTION

In computer vision, research is undergoing for artificial systems to extract information from images. This will allow system to “see” and “understand” their environments. Despite many advances, artificial systems still fall short comparing to their biological counterpart. One of these systems, automated target recognition system (ATR), is in development for human-like performance to locate and target any object or feature in a given image or video. NASA/JPL has developed a multi-stage ATR system, which compose of a Grayscale Optical Correlation (GOC) filter on 512 x 512 pixel images for target detection adaptive neural network (NN)<sup>1</sup>. The multi-stage system finds a good balance between speed and accuracy. The first stage GOC can detect and locate ROI of input scene relatively quickly, but with high false-alarm rate. Thus the second stage NN checks the ROI for verification on true-positive targets. The two-stages ensure speed by GOC and accuracy by NN<sup>2</sup>.

## 2. OT-MACH CORRELATION FOR TARGET DETECTION

GOC uses the Optimum Trade-off Maximum Average Correlation Height (OT-MACH) algorithm<sup>3</sup>, as in Eq. (1), for optimal scan of images to find locations of “regions-of-interest” (ROIs) that may contain potential targets. OT-MACH performs the operations in the Fourier domain and finds correlations in an optical correlator<sup>4</sup>. The OT-MACH algorithm is created and optimized in a computer<sup>5</sup>, and this simulation simultaneously balanced several conflicting performance measures: Output Noise Variance (ONV), Average Correlation Energy (ACE), Average Similarity Measure (ASM), and Average Correlation Height (ACH)<sup>3</sup>. The filter is created to minimize the function:

$$E(h) = \alpha (\text{ONV}) + \beta(\text{ACE}) - \gamma(\text{ACH}) \quad (1)$$

---

<sup>1</sup> e-mail: Thomas.T.Lu@jpl.nasa.gov , Tel: (818) 354-9513, Fax: (818) 393-4272

The coefficients  $\alpha$ ,  $\beta$  and  $\gamma$  control properties of the filter such as noise and distortion tolerance; thus they must be optimized to create the best filter for a set of images. The algorithm finds the optimized values for coefficient  $\alpha$ ,  $\beta$  and  $\gamma$  for various application, and in this paper the coefficients are optimized for finding small boats. Selections for coefficients  $\alpha$ ,  $\beta$  and  $\gamma$  are automated in a program to approach optimized OT-MACH performance in as little iteration as possible. The program approaches the optimum using adaptive step gradient descent algorithm<sup>5</sup>. Before performing gradient descent, the program performs permutation to determine rough values of  $\alpha$ ,  $\beta$  and  $\gamma$  to minimize only converging to local maximum instead of a global maximum. The final values of the coefficients is determined based on performance metrics of Correlation Peak Height (PK) and Peak-to-Side lobe (PSR) Ratio, in which the ratio is correlated with the filter’s target detection in true positive rate and false positive reduction<sup>4</sup>.

### 3. TRAINING OT-MACH FILTER

OT-MACH filter is a Fourier transform based filter, making it shift invariant, but sensitive to scale, rotation, perspective, and lighting conditions. Training images must provide representative samples to cover all the variation required for a general representation. Training sets are selected from different videos to represent the different angles and sizes of the boats. Some examples of the training boats with different sizes are shown in Fig. 1. Figure 1(e) shows a challenging image of a small boat in a cluttered shore line background. The task is to identify the boat as far away as possible without generating too many false alarms.

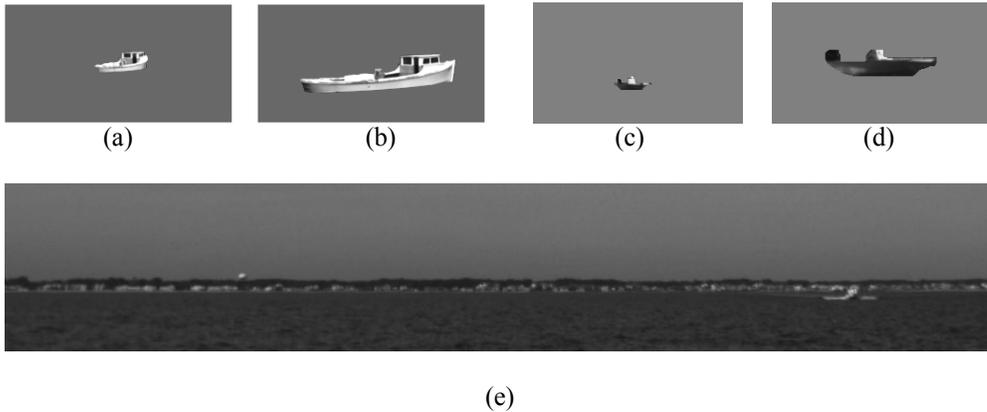


Figure 1: Examples of training set for OT-MACH. Training set must have taken account of different perspectives, sizes, and lighting conditions. Here we have (a) medium and (b) large size of white boats; and (c) medium and (d) large size of black boats; (e) an image (right side) of a boat against a shore line background.

Besides training sets including different sides, sizes, and lighting conditions, all image trainings sets must be resized to 512x512 pixels, accepted by the OT-MACH filter. In this paper, about 5-10 boat models are used for training and 3-5 boat images are used for testing the filters. The permutation and gradient descent are ran to find the optimal values for  $\alpha$ ,  $\beta$  and  $\gamma$ . In the first stage of the ATR process, the OT-MACH identifies many ROIs in the image, as shown in Fig. 2. Any object similar to the potential targets is marked with a red box (ROI). The ROIs are sent to the next stage for verification.

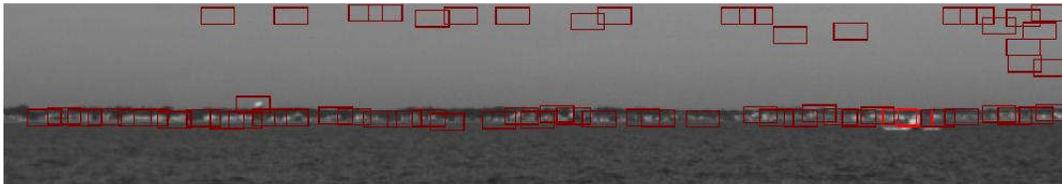


Figure 2: Images processed after OT-MACH filter. Potential “Regions-of-interest” are boxed with red boxes, with the desired target in purple dot. Each “Region-of-interest” is further processed and classified to determine if the “Region-of-interest” contains a target.

#### 4. FEATURE EXTRACTION WITH PRINCIPAL COMPONENT ANALYSIS

The ROIs identified by the GOC stage represent small windows which contain possible target in the image. The dimensionality of these regions in pixel space is the area of the window of red boxes in Fig. 2. The data contained in the small window of ROI is too large for input into a neural network, and thus smaller but unique data for each ROI must be extracted for classification. In order to retain the information about target but in smaller set of pixel data, feature extraction technique must be applied on each ROI. Pixel data from each ROI may also contain “noise,” non-boat data such as waves or background, and thus training sets are picked carefully with the least “noise”. Principal Component Analysis (PCA) algorithm<sup>6</sup> is used to extract features from ROI, with the top 18 vectors output from PCA used as the features for each ROI.

#### 5. NEURAL NETWORK LEARNING

After PCA feature extraction, we use a backpropagation feedforward neural network (BFNN) with Levenberg-Marquardt algorithm<sup>7</sup> to train and classify the ROIs. In this paper two neural networks are compared, naming Radial Basis neural net<sup>8</sup> (RBNN) and BFNN.

##### 5.1 Backpropagation Feedforward Neural Network (BFNN) Training

In a BFNN, we have input and output pairs  $\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_N, t_N\}$ , where  $p_N$  are the inputs, in this case, the feature vectors from PCA; and  $t_N$  are the corresponding targets, in this case we set boat targets as 1 and the other non-targets as 0. A multi-layer BFNN is illustrated in Fig. 3. Each layer takes inputs and multiplies it by weight matrix  $\mathbf{W}$  and add bias  $b$  to it, shifting the multiplication according to the bias  $b$ . Then the outputs are sent to transfer function for a new mapping of the results. Both layers are mapped between -1 and 1 through a tan-sigmoid transfer function. The final output vector  $a$  is shown Eq. (2).

$$a^1 = \text{tansig}(W_1 p + b_1) \quad a = \text{tansig}(W_2 a^1 + b_2) \quad (2)$$

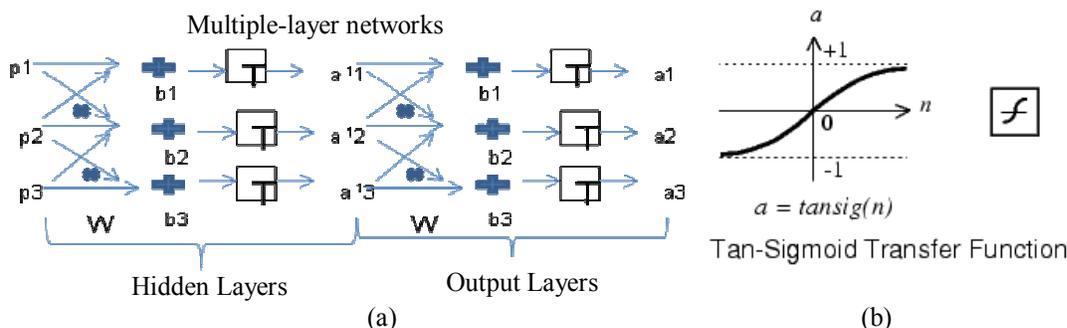


Figure 3: Backpropagation Feedforward Multi-layer networks (a) with its transfer function (b) in both layers.

Next, the weights in the layers are adjusted according to how closely output  $a$  correspond to target  $t$  through a performance function. Here we use Mean squared error with regularization as the performance function. Mean squared error  $mse$  is defined as in Eq (3).

$$F = mse = \frac{1}{N} \sum_{i=1}^N (e_i)^2 = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2 \quad (3)$$

Where  $N$  is the number of inputs,  $t$  the target vector and  $a$  the output vector from the neural network. Regularization will improve the generalization of the network by modifying the performance function by adding a term that consists of the mean of the sum of squares of the network weights and biases, as in Eq (4).

$$msereg = \gamma mse + (1 - \gamma) msw \quad (4)$$

Where  $\gamma$  is the performance ratio and sum of squares of the network weights and biases  $msw$  as Eq (5).

$$msw = \frac{1}{n} \sum_{j=1}^n w_j^2 \quad (5)$$

The neural network updates its weight according to Levenberg-Marquardt algorithm, as in Eq (6). This algorithm is a variant of the Quasi-Newton optimization method.

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{e} \quad (6)$$

Where  $e$  is the error vector, and  $J$  is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, in which the Jacobian matrix for single neuron can be written as follows:

$$J = \begin{bmatrix} \frac{\partial \varepsilon_1}{\partial w_1} & \dots & \frac{\partial \varepsilon_1}{\partial w_n} & \frac{\partial \varepsilon_1}{\partial w_0} \\ \vdots & & \vdots & \vdots \\ \frac{\partial \varepsilon_p}{\partial w_1} & \dots & \frac{\partial \varepsilon_p}{\partial w_n} & \frac{\partial \varepsilon_p}{\partial w_0} \end{bmatrix} = \begin{bmatrix} x_{1_1} & \dots & x_{n_1} & 1 \\ \vdots & & \vdots & \vdots \\ x_{1_p} & \dots & x_{n_p} & 1 \end{bmatrix} \quad (7)$$

Where  $w$  is the vector of the weights,  $w_0$  is the bias of neuron, and  $\varepsilon$  is the error vector. The  $\mu$  is a scalar value that makes sure the performance function is always reduced in each iteration of the algorithm. When the scalar  $\mu$  is zero, the performance function is just Newton's method, using the approximate Hessian matrix  $H=J^T J$ . When  $\mu$  is large, performance function becomes gradient descent with a small step size. The idea is to shift toward Newton's method near an error minimum, because Newton's method is faster and more accurate near an error minimum. Thus,  $\mu$  is decreased after each successful step, which is a reduction in performance function, or increase otherwise to increase the performance function.

Another method used to improve generalization is “early stopping”, which is to prevent overfitting. Input data is divided into three subsets, training set for network training, validation set, and testing set. In *early stopping* testing set is not used. During training both training error and validation error would decrease. However, when the network starts to overfit the data, validation will begin to rise. After validation error rises for a few iteration, training is stopped to prevent overfitting, and weights and biases at the minimum validation error are used. All the ROIs used for training are divided randomly in three sets for *early stopping*, with 60% of the samples used for the training set, 20% for the validation set, and 20% for the test set.

## 5.2 Radial Basis Neural Network (RBNN) Training

RBNN also has a multiple-layer structure, using radial basis transfer function for the first layer and liner transfer function for the second layer, as shown in Fig. 4.

Again we have input and output pairs  $\{p_1, t_1\}, \{p_1, t_1\}, \dots, \{p_N, t_N\}$ . However, RBNN maps the input and output differences based on Eq. (8)

$$a^1 = radbas(\text{dist}|W_1 - p|*b_1); \quad a = linear(W_2 a^1 + b_2) \quad (8)$$

Radial Basis transfer function calculates the distance between inputs and weights, and does element-by-element multiplication with the biases. The linear layer acts the same way as the BFNN, with inputs multiplied by the weights and shifted by biases. Radial Basis transfer function is a Gaussians, defined by Eq. (9).

$$radbas(n) = e^{-n^2} \tag{9}$$

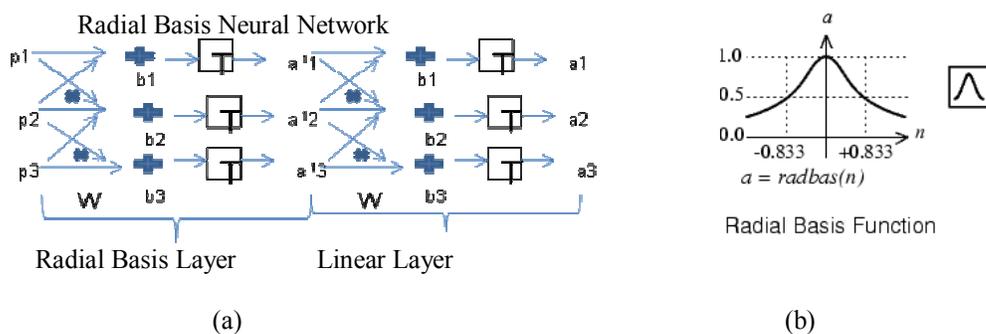


Figure 4. Radial Basis Neural Network (a) and Radial Basis transfer function (b).

Each bias in the Radial Basis layer is set to  $\sigma/SPREAD$ , where  $\sigma$  is a spread constant, usually  $\sigma = 0.8326$ . The SPREAD variable is generally proportional to the half-width of the Gaussians. Thus, within  $\pm SPREAD$ , radial basis function would output greater than 0.5. This determines the width of an area in the input space to which each neuron responds. Each radial neuron will respond with 0.5 or more to any input vectors within a vector distance of SPREAD from their weight vector, and less than 0.5 if the distance is greater than SPREAD. For a good generalization, SPREAD should be large enough that neurons respond strongly to overlapping regions of the input space, but SPREAD should be smaller than the distance across the whole input space.

The SPREAD value, mean square goal error, and maximum number of neurons allowed are given before running RBNN. The algorithm then creates one neuron at a time. At each iteration, the network is simulated, and the input vector that lower the mean square error function between the targets and neural outputs the most, is used for creating new radial neuron. The network is simulated again. If the goal error is reached or maximum number of neurons is reached, radial basis is complete. Otherwise repeat the same process.

## 6. EXPERIMENTAL RESULTS: COMPARISON BETWEEN RBNN AND BFNN

After feature extraction, a feature must be classified as either a target of small boat or a false target. Here neural networks were used to classify the features from the ROIs, with output value close to 1 as true target and 0 as false targets. ATR has been tested with both RBNN and BFNN with underwater targets and jets <sup>2,6</sup>. Both methods show promising results. Here we will compare the two neural networks with four video sets and a combined set with all video that were used in the experiment. In total, more than 50,000 ROI, or hits for the images, are classified. The numbers of training and total hits are in Table I. Training sets are kept to minimum but representative. Notice true hits used for training are from 14% to 30% maximum. False hits used for training are very low, up to 5% maximum.

Table I. Training Set used for Neural Network Training

Sample Set	Total Training samples / all hits	Total true hits / all hits	% of True hits used for training / all true hits	False hits used for training / all false hits
Oct22-12	265 / 27493 = .96%	286/27493 = 1.04%	42 / 286 = 14.69%	223 / 27207 = .82%
Sept17	538 / 9362 = 5.75%	263 / 9362 = 2.81%	76 / 263 = 28.90%	462 / 9099 = 5.08%
Oct22_11	193 / 11433 = 1.69%	91 / 11433 = 0.8%	25 / 91 = 27.47%	168 / 11342 = 1.48%
06-09_cam3	209 / 4682 = 4.46%	139 / 4682 = 2.97%	43 / 139 = 30.94%	166 / 4543 = 3.65%
All Videos	1205 / 52970 = 2.27%	779 / 52970 = 1.47%	186 / 779 = 23.88%	1019 / 52191 = 1.95%

The parameters set for Radial Basis (RBNN) and Backpropagation Neural Networks (BPNN) are shown in Table II.

Table II: Parameters for the Neural Networks

Sample	Total Neurons	Mean Square Error Goal	Spread (Radial)	Mean distance among all points (Radial)	Number of Runs
Oct22-12 - RBNN	80	0.004	14.1234	44.1345	20
Oct22-12 - BPNN	40	.001	N/A	N/A	50
Sept17 - RBNN	225	0.004	20.4272	41.6882	20
Sept17 - BPNN	30	.001	N/A	N/A	50
Oct22_11 - RBNN	25	0.004	27.1627	38.8038	20
Oct22_11 - BPNN	30	.001	N/A	N/A	50
06-09_cam3 - RBNN	80	0.004	15.6301	44.6575	20
06-09_cam3 - BPNN	30	.001	N/A	N/A	50
All Videos - RBNN	380	.004	21.1331	43.043	20
All Videos - BPNN	40	.001	N/A	N/A	50

Figure 5 shows a BPNN identified the boat correctly and also eliminated all of the false alarms generated by the first-stage OT-MACH filter correlation operations as shown in Fig. 2.



Figure 5. A RBNN correctly identified the boat (marked in a box) and eliminated all of the false alarms.

The number of neurons for RBNN is determined by adding neurons one at a time until the networks reach mean square goal error 0.004. BPNN uses 30 - 40 neurons, which are enough for good results.

Goal errors for RBNN are set as 0.004, which is a good threshold for RBNN to have enough neurons to generalize and not overfitting the classification. BPNN goal errors are set as 0.001 to achieve good accuracy and generalization.

### 6.1 RBNN Spread Optimization

SPREAD is not apparent to determine for RBNN. Here we first obtain average mean of the distances among every pair of input points, and with this distance we test different percent of it to set the SPREAD. For each SPREAD value we see how it performed based on Frequency Relative Operating Characteristic (FROC) curve, plotting the average false positives per image against the True Positive Rate (TPR). The performance is the area under the FROC curve from 0 false positive to 10 false positives. This will give us a good representation of the highest accuracy with lowest average false positives hits. Example of FROC results for all combine videos in RBNN is shown in Fig 6.

Initially SPREAD is set as 25% of mean distance, and with step size of 5% increment, we tested all the SPREAD value up to 70% of mean distance. Next, whichever percentage resulted in the highest score, we again tested from -5% to +5% of the percentage with 1%-2% step size. This testing allows us to find the optimal SPREAD and avoid reaching local maximum value with gradient descent.

We tested SPREAD performance from 0% to 150% of the mean distance among all inputs, as shown in Fig. 7. As can be seen, with the exception of data Oct22-11, all of the video sets perform near optimal between 30% to 70% of mean distance. Although choosing the SPREAD value is not apparent, SPREAD value can be narrowed down in some range based on mean distance of all inputs.

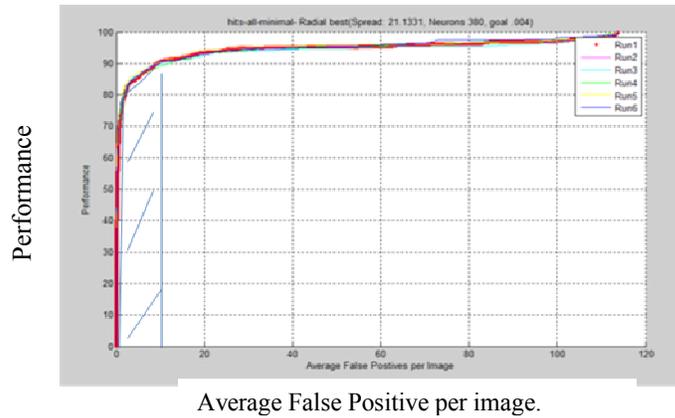


Figure 6. FROC curve for measuring performance of neural networks. Area under the curve with average false positives per image from 0 to 10 is used as performance measure.

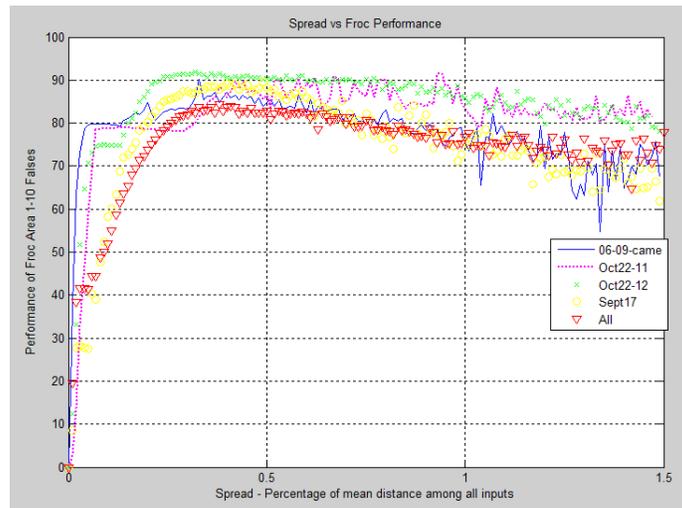


Figure 7. Spread values vs. Performance. Spread is determined by percentage of mean average of distance among all inputs. Spread value performs near optimal between 0.3 to 0.7.

The performance comparison of the RBNN and BFNN is shown in Fig. 8. As can be seen the BFNN performs slightly better with video sets Oct22-12 and Sept 17, while Radial Basis performs better with Oct22-11, 06-09, and with all video sets. The video sets where Radial Basis performs better has more “noise” in the background, as in Oct22-11 the boat seen along with shoreline building in similar white color, and in 06-09 the boats have front and side shots with different lighting condition.

### 6.2 Number of Neurons Optimization

The number of neurons in RBNN also has an effect on its performance, training and classification time. In Fig. 8, four video sets are tested. As can be seen, having more neurons will improve performances until reaching a limit, and the performance will start leveling off or actually will decrease. With more neurons Radial Basis neural network will start overfitting the data, and instead of generating a general network the network will memorize the data. In the Radial Basis neural network implementation, the algorithm continues to add neurons until the goal error level is reached, which will not help neural network performance as shown in. As shown in Fig. 9, a better way to decide for optimal number of neuron is check against the FROC curve performance. If after a few iterations the FROC curve is

starting to level off or decreasing, it would be better to stop adding neurons to preserve generalization. Adding more neurons will also incur the costs of training and classification time. From the test results in Fig. 9, although classification time only grows linearly, but training time grows exponentially which may be too costly to use in some systems.

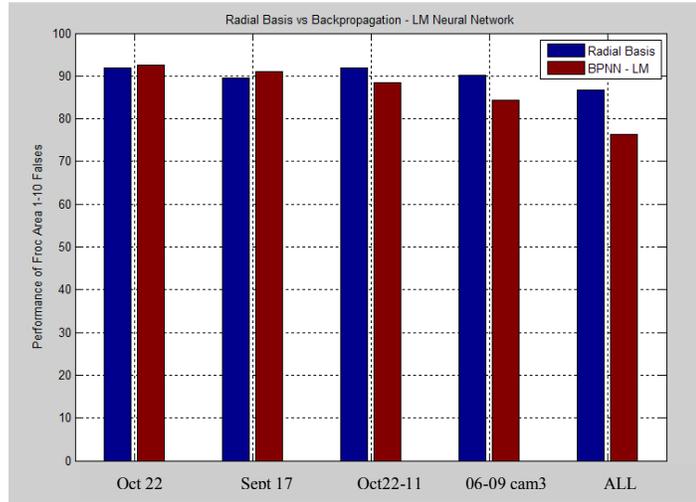


Figure 8: Performance of Radial Basis (Blue) and backpropagation neural network (Red) trained and classified on each individual set. Radial Basis performed better in very “noisy” video sets, as seen from set ‘06-09 cam3’ and ‘ALL’.

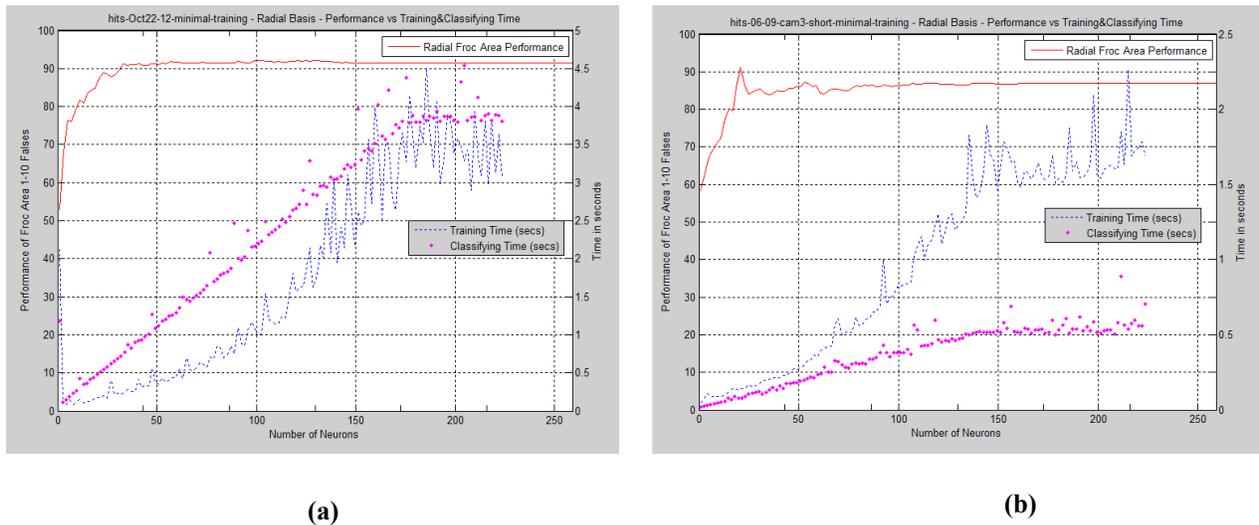


Figure 9: Number of Neurons affect on performance, training and classification time. Two data sets are shown in (a) and (b). The performance is indicated by red solid line, training time by dotted blue line, and classification time by purple thick dotted line. From each video set, adding more neurons did not help performance but resulted in constant or actual decrease in performance, while classification time grew linearly but training time grew exponentially.

Here we compare the two neural networks on training and classification time, as shown in Fig. 10. The training time indicated is the total training time to reach the optimal network status. We train the RBNN with 20 runs, first 10 times with step size of 5 and second 10 times with step size of 1. The BFNN is trained by running 50 times. As can be seen from training with all video hits, there is a huge cost of time with Radial basis with many neurons. The BFNN training time would increase if the images are very noisy, which would require the learning algorithm to take

longer time to reach lower error. For classification time, we show the classification time per hits in the videos. The BFNN classification time is consistent, while the Radial Basis classification time depends upon how many neurons are in the network.

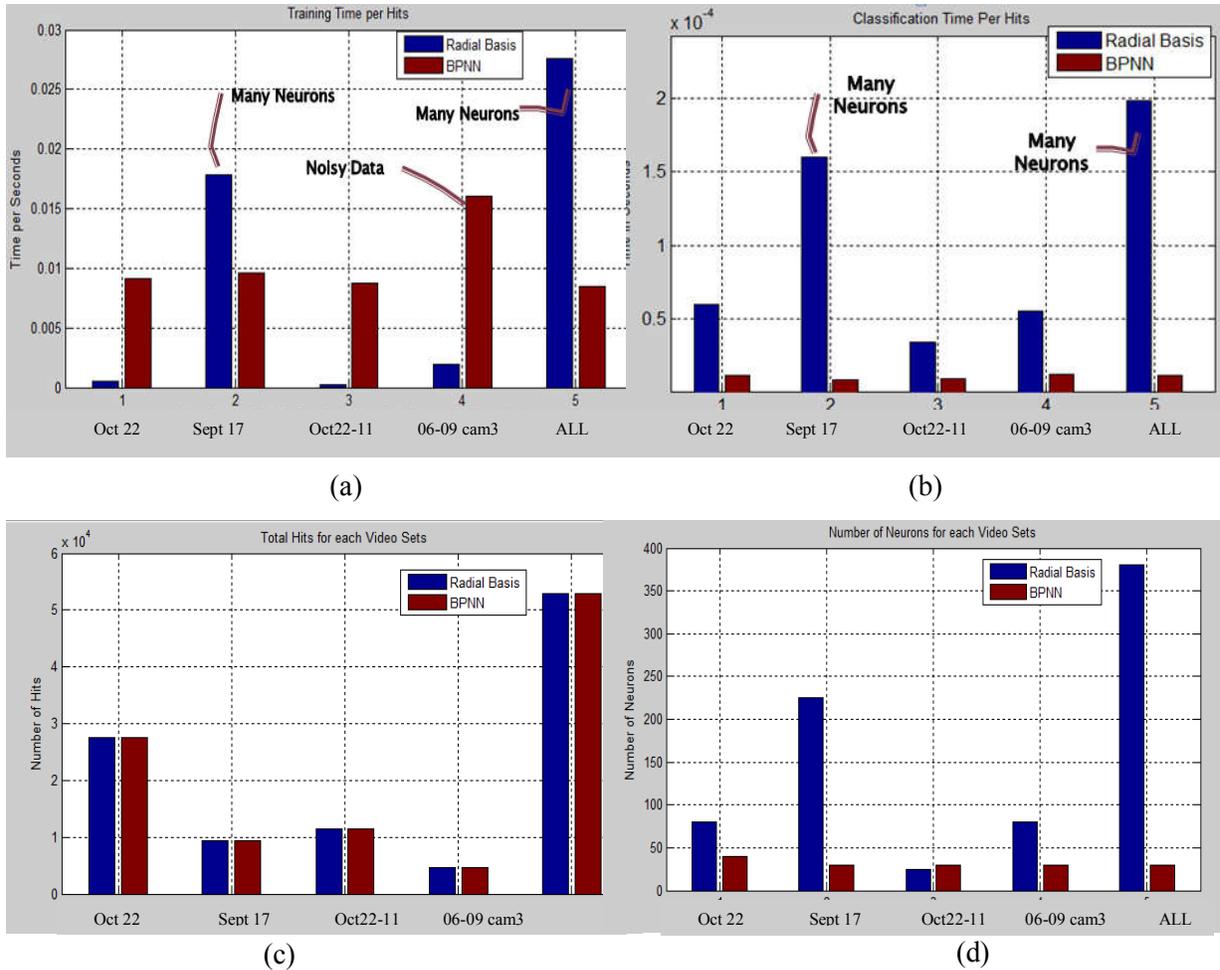


Figure 10. Classification (a) and training (b) time per hits for each video set. Radial Basis classification and training time depended heavily on number of neurons. Backpropagation network classification time was consistent, and training time depended on how noisier is the data.

Finally, to test how each neural network generalize, we train one general network for all the video sets and classify on each individual video, as shown in Fig. 11. Radial Basis performs slight better in two of the videos and much better on one video. This might be because the video is noisier, with different lighting condition and has front and side view of the boats. The BFNN performs nearly as well as Radial Basis, and classifies faster than the RBNN.

## 7. CONCLUSION

Comparison has been shown between Radial Basis and backpropagation neural network. The RBNN has the ability to deal with noisy images and may be more consistent in catching small boats due to its nature of “sphere of influence” to classify inputs not seen before. However, the RBNN suffers from large number of neurons to run,

making the BFNN faster to train and classify, and in some cases perform better than the RBNN. Future work can be directed in speeding up the RBNN by using necessary number of neuron to classify the objects. The BFNN can also be further optimized for better performance. Both networks offer different advantages and whichever one is more suitable depends on the system requirement for speed or accuracy.

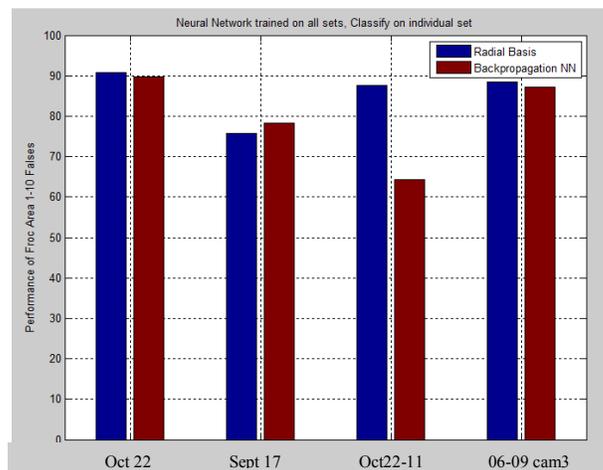


Figure 11. Performance classified by one optimal Radial Basis neural network and one optimal Backpropagation neural network trained on all video sets. Radial Basis performed much better in video set Oct22-11, which was very noisy. Other video sets Radial Basis and Backpropagation neural networks performed nearly the same.

## 8. ACKNOWLEDGMENTS

This research was carried out at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration (NASA). We also acknowledge the support by the Undergraduate Student Research Program (USRP) through NASA.

## REFERENCES

- [1] Lu, T., Hughlett, C., Zhou, H., Chao, T. H., Hanan, J. C., "Neural network postprocessing of grayscale optical correlator," Proc. of SPIE, 5908, (2005).
- [2] Ye, D., Edens, W., Lu, T., Chao, T., "Neural Network target identification system for false alarm reduction," Proceedings SPIE Optical Pattern Recognition XX, 7340, (2009).
- [3] Zhou H., Hughlett C. L., Hanan J. C., Lu T., Chao T-H. "Development of streamlined OT-MACH-based ATR algorithm for grayscale optical correlator," *SPIE Optical Pattern Recognition XVI*; **5816**, pp. 78-83, (2005).
- [4] Chao, T., Zhou, H., and Reyes, G., "Compact 512x512 Grayscale Optical Correlator", SPIE vol. 4734, p. 9-12, (2002).
- [5] Johnson, O.C., Edens, W., Lu, T., Chao, T., "Optimization of OT-MACH filter generation for target recognition," Proceedings SPIE Optical Pattern Recognition XX, 7340, (2009).
- [6] Shlens, J., "A Tutorial on Principal Component Analysis," Institute for Nonlinear Science, University of California, San Diego, La Jolla, CA 92093.
- [7] Hagan, H.B. Demuth, and M.H. Beale, "Neural Network Design", Martin Hagan, (2002).
- [8] Chen, S., C.F.N. Cowan, and P.M. Grant, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," IEEE Transactions on Neural Networks, 2, No. 2, p. 302-309, (1991).